Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Title: FCFS Disk scheduling

Submitted To:	**Submitted By:**
Atia Rahman Orthi Lecturer Department Of Computer Science & Engineering	Jannatun Saumoon ID: 21201066 Sec: B2

Problem Statement:

Input: Req Seq= {176,79,41,11}

Head:50

Output: 291

Steps:

1. Define a function to calculate total seek time.
2. Initialize seek_total and head_position.
3. Loop through each disk request:
    - Calculate the absolute difference between current head and request.
    - Add it to seek_total and update the head position.

4. Return the total seek time.
5. Input the request sequence and initial head position.
6. Call the function and print the result.

Source Code:

FCFS_Disk_scheduling

```python
def calculate_fcfs_seek_time(track_requests, start_head):
    seek_total = 0
    head_position = start_head

    for track in track_requests:
        distance = abs(track - head_position)
        seek_total += distance
        head_position = track

    return seek_total


# Input data
disk_requests = [176, 79, 41, 11]
initial_position = 50

# FCFS scheduling execution
total_seek = calculate_fcfs_seek_time(disk_requests, initial_position)

print("Total Seek Time (FCFS):", total_seek)
```

Output:

```
Total Seek Time (FCFS): 291
```

Conclusion:

The FCFS disk scheduling algorithm processes requests in the order they arrive, adding the absolute distance between the current and next request. It is simple, fair, and easy to implement, making it suitable for basic applications. However, FCFS can result in high total seek time, especially when requests are widely scattered, making it less efficient compared to algorithms like SSTF or SCAN. This inefficiency becomes more noticeable in high-volume systems where performance is critical. Despite its drawbacks, FCFS remains a foundational algorithm in disk scheduling, offering a basis for understanding more advanced, optimized approaches.